MACM 316 – Computing Assignment 4

Due Date: February 28 at 11:00pm.

Submission Instructions: You must upload one .pdf file in Crowdmark that consists of two pages: page 1 is your report which should fit all discussions, data and figures into a single page; and page 2 is a listing of your code. The deadline is **11:00pm** on the due date. The actual due time is set to 11:05pm and if Crowdmark indicates that you submitted late, you will be assigned a grade of 0 on this assignment. Your TA has emailed you a Crowdmark link that you should save since it will allow you to upload your completed assignments.

- Please review the **Guidelines for Assignments** carefully.
- Acknowledge any collaborations or assistance from colleagues/TAs/instructor.
- If you have any questions about Matlab or aspects of this assignment, then you are strongly encouraged to attend tutorials and drop-in workshops.

Computing Assignment – Gaussian Elimination on Large Random Matrices

To complete this assignment, you should start by downloading the `GERandom.m` from Canvas, under "Computing Assignments".

In class you have seen that finite-precision computations with Gaussian Elimination (such as that implemented in Matlab's backslash command) generates relatively small errors for small matrices like those we considered in lectures, but the error can grow much larger for when the matrix is larger. The purpose of this assignment is to quantify the growth of this error for large matrices that have random entries.

Let A be an $N \times N$ matrix with random entries sampled from a uniform distribution on $[-1, 1]$. The exact solution is set to $x = (1, 1, \ldots, 1)^T \in \mathbb{R}^N$ and then $b = Ax$ is taken to be the corresponding right-hand side vector. The linear system $Ax = b$ is then solved numerically with Matlab's backslash operator, using the command `y = A \ b`, which yields an approximate solution $y = [y_i] \in \mathbb{R}^N$. To measure the error between the exact solution x and the approximation y, we define

$$\epsilon = \max_{1 \leq i \leq N} |x_i - y_i|$$

to be the maximum component-wise difference between the two vectors (this is just the max-norm error). Since A is a random matrix, you need to run this calculation for a number of separate trials using different realizations of A in order to obtain a reasonable averaged value for ϵ. Let M be the number of trials and suppose that for the k^{th} trial the error is $\epsilon^{(k)}$. Then you can define the mean (average) error as

$$E_N = \frac{1}{M}\left(\epsilon^{(1)} + \epsilon^{(2)} + \cdots + \epsilon^{(M)}\right)$$

where the subscript N on E_N denotes the matrix size. The code `GERandom.m` provided to you computes this averaged error E_N for a collection of M random matrices all of size N.

The goal of this assignment is for you to study the growth in the error E_N as N increases, and to estimate the size of the matrix $N = N^*$ for which the mean error in Gaussian Elimination reaches $E_N \approx 1$. In other words, you need to find the size N^* for which the round-off error will grow to the same magnitude as the solution vector x (i.e., <u>no</u> significant digits of accuracy remain).

In practice, N^* is quite large and your computer will not have the processing power to find N^* directly. Instead, you should extrapolate the data from your computations on a sequence of moderate-sized N values in order to estimate N^*. To do so, generate a plot of $\log(E_N)$ versus $\log(N)$ and then extend the data to $E_N = 1$ using suitable extrapolation. Your conclusions should be explained in a one-page report which includes the following:

- A justification for the values of N and M that you chose.
- Your plot of $\log(E_N)$ versus $\log(N)$.
- An explanation of how you do the extrapolation.
- An estimate of the size N^* where you predict that the error $E_{N^*} \approx 1$.

You should find the following Matlab built-in functions useful:

- `A = 2*rand(N,N)-1` : this command returns an $N \times N$ matrix with random entries chosen from the interval $[-1, 1]$.
- `loglog(Nlist, Elist)` or `plot(log10(Nlist), log10(Elist))` : generates a log-log plot of the errors (`Elist`) versus the corresponding matrix sizes (`Nlist`).
- `p = polyfit(x, y, 1)` : for a set of data points (x, y), this returns the coefficients of a straight-line fit having the form $y = p_1 x + p_2$. Replacing the last parameter with an integer greater than 1 fits a higher degree polynomial.